Dynamically Scalable Fog Architectures

Dominic Henze

Abstract Recent advances in mobile connectivity as well as increased computational power and storage in sensor devices have given rise to a new family of software architectures with challenges for data and communication paths as well as architectural reconfigurability at runtime. Established in 2012, Fog Computing describes one of these software architectures. It lacks a commonly accepted definition, which manifests itself in the missing support for mobile applications as well as dynamically changing runtime configurations. The dissertation "Dynamically Scalable Fog Architectures" provides a framework that formalizes Fog Computing and adds support for dynamic and scalable Fog Architectures.

The framework called xFog (E**x**tension for **Fog** Computing) models Fog Architectures based on set theory and graphs. It consists of three parts: xFogCore, xFogPlus, and xFogStar. xFogCore establishes the set theoretical foundations. xFogPlus enables dynamic and scalable Fog Architectures to dynamically add new components or layers. Additionally, xFogPlus provides a View concept which allows stakeholders to focus on different levels of abstraction.

These formalizations establish the foundation for new concepts in the area of Fog Computing. One such concept, xFogStar, provides a workflow to find the best service configuration based on quality of service parameters.

The xFog framework has been applied in eight case studies to investigate the applicability of dynamic Fog Components, scalable Fog Architectures, and the service provider selection at runtime. The case studies, covering different application domains—ranging from smart environments, health, and metrology to gaming—successfully demonstrated the feasibility of the formalizations provided by xFog, the dynamic change of Fog Architectures by adding new components and layers at runtime, as well as the applicability of a workflow to establish the best service configuration.

D. Henze (✉)
Technical University of Munich, Munich, Germany
e-mail: henzed@mytum.de

M. Felderer et al. (eds.), *Ernst Denert Award for Software Engineering 2020*,
https://doi.org/10.1007/978-3-030-83128-8_6

1 Introduction

With the constant increase of computational power and available storage, mobile devices get more and more involved in distributed systems, which are "collections of independent computers that appear to be one single system to users" [1]. Nevertheless, mobile devices will always be resource poor in comparison to static hardware, as static hardware is not capped by properties such as heat dissipation or battery life [2, 3]. Therefore, mobile devices will always struggle with the most advanced media and data analysis.

Mobile cloud computing was introduced to bridge this gap and combines mobile computing with cloud computing to leverage the computational power of the cloud for mobile devices [4, 5]. However, clouds are usually distant from the mobile devices and using them creates high latencies, which are insufficient for real-time applications such as augmented reality.

To address this issue, concepts such as Cloudlets, Edge Computing, and Fog Computing emerged. Satyanarayanan et al. described these concepts to utilize resource-rich components near the mobile device to offload computational intense tasks while having "low latency, one-hop, high-bandwidth wireless access" [3]. While Cloudlets use trusted, nearby components with excessive computational power, Edge Computing focuses on the entirety of the network trying to push services as close to the edge as possible [6, 7].

Bonomi et al. introduced Fog Computing as a three-layered software architecture containing a Cloud, Fog, and Edge layer [8]. These layers interact using subscriber models with one layer acting as the provider and the other one as its user. Accordingly, application scenarios such as dynamic vehicles, smart grids, distributed sensor networks, and smart environments can benefit from using Fog Computing.

This loose definition has led to many interpretations of Fog Computing as well as attempts to sharpen the definition. Nevertheless, there is no commonly accepted definition of what Fog Computing or a Fog Node is, and the difference to similar concepts such as Edge Computing is not clearly defined [9].

To address these misunderstandings, the dissertation "Dynamically Scalable Fog Architectures" [10] follows the intent of the paper "Fog horizons—a theoretical concept to enable dynamic fog architectures" [17] to create a formalized definition for Fog Computing based on software architectures and set theory. It is based on the organized and systematic research approach design science established by Hevner and Wieringa [11, 12]. It investigates the following knowledge and technical research goals:

> **Knowledge Goal 1:** Establish Fog Computing as a subclass of software architectures.

> **Technical Research Goal 1:** Define a framework that provides a formalized definition of Fog Computing.

These goals are achieved with xFog, an e**x**tension for **Fog** Computing, and xFogCore which defines the foundations of the framework. These foundations are then used to address two additional challenges by formalization: Providing support for dynamic components and addressing the ambiguity of layers, in particular of the Fog.

> **Technical Research Goal 2:** Define an extension to the foundations of Fog Computing that supports components joining and leaving an architecture (*Dynamics*).

> **Technical Research Goal 3:** Define an extension to the foundations of Fog Computing that supports the process of adding and removing layers (*Scalability*).

These goals should be addressed with xFogPlus, one part of xFog that relies on the foundations introduced by xFogCore and formalizes dynamics and scalability in Fog Architectures.

The combination of both, xFogCore and xFogPlus, enables xFog to support a variety of advanced concepts, such as xFogStar. xFogStar is an extension that investigates the selection of service providers which is addressed with the following goal:

> **Technical Research Goal 4:** Enable service provider selection in dynamic scalable Fog Architectures.

Finally, we want to investigate if and how xFog, its parts, and extension describe Fog Computing:

> **Knowledge Goal 2:** Investigate the feasibility of xFog.

> **Knowledge Goal 3:** Investigate the feasibility of xFogStar.

Therefore, we validate three different aspects: *Dynamic Fog Components* , *Scalable Fog Architectures* , and *Service Provider Selection*. Each aspect is addressed by multiple case studies with cases from different domains.

2 xFog: An Extension for Fog Computing

To define an architecture based on the component and connector definitions by Shaw and Garlan, Bass and Kruchten for software architectures [13–15], we have to investigate the components included within an architecture and the connections

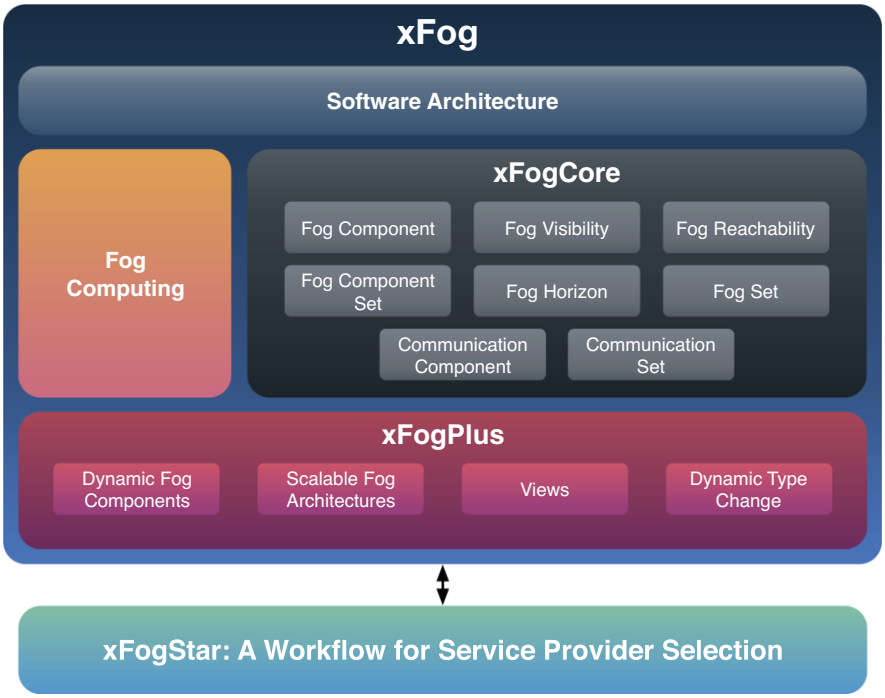

Fig. 1 Overview of the xFog framework highlighting the most important concepts

among them. The framework xFog addresses this goal for Fog Computing and its resulting architectures for the purpose of formalization. The name xFog is short for an e**X**tension for **Fog** Computing and accordingly uses its principles and ideas to provide a formalization which addresses knowledge goal 1 and the technical research goal 1 (Fig. 1).

xFog is separated into two parts: xFogCore and xFogPlus. While xFogCore focuses on the formalization of Fog Computing, xFogPlus extends these formalizations by the concepts of dynamic reconfigurability and scalability. Finally, xFogStar extends xFog with a workflow that supports service consumers with the selection of service providers.

2.1 Fog Component

Fog Computing is defined using three types of components: *Cloud Devices*, *Fog Nodes*, and *Edge Devices*. While Cloud Devices are servers / data centers offering storage, computational power, or specific software, Fog Nodes are devices on the way between the cloud devices and edge devices that could potentially offer services

with less computational power but faster response times, location awareness, and mobility [8]. Lastly, Edge Devices are end-user devices that strive to use services offered by cloud devices and/or Fog Nodes.

In order to make the following concepts more type independent and easier to understand, we define a *Fog Component* as a common superclass. This Fog Component shows the direct similarity to the software architecture definitions and allows its comparability. The group of all Fog Components is called Fog Component Set.

2.2 Fog Visibility

The first concept based on the Fog Component definition is the Fog Visibility. The Fog Visibility is the virtual area around a Fog Component including all other Fog Components that can be "seen." This idea is translated to the terminology of software architectures and networking as shown in Definition 1 using received messages without transitivity.

Definition 1: Fog Visibility

$FogVisibility(x) := \{y \mid y \text{ receives direct messages from } x\}$

with:
$x, y \in FogComponentSet$

Therefore, the Fog Visibility describes a relation on the Fog Component Set which allows us to use and assign properties of sets and relations to it.

One limitation of the Fog Visibility and all following concepts is the communication channel used by the Fog Components. While communication channels such as Wi-Fi, 3G, or 4G can potentially include a large amount of Fog Components, others, such as Bluetooth, might be inherently limited. This aspect will be further investigated in Sect. 2.7.

Figure 2 shows two examples of Fog Visibilities for a Fog Component A. The Fog Visibility on the left describes the self-containing set and the Fog Visibility on the right a more generic example. To show the Fog Visibility in both cases, we display the range in which a Fog Component can send messages as circles. This shape as well as the radius of the Fog Visibility can change substantially depending on the communication channel used.

Also the first example is of less practical use; it describes the edge case of the Fog Visibility definition and answers the question if Fog Components without other communication partners still maintain Fog Visibilities. As we do not exclude the Fog Component itself as a viable communication partner, the Fog Visibility relation is reflexive and, therefore, exists even without other Fog Components in range.

The second example includes Fog Components A - F with Fog Components A - C and E being inside A's Fog Visibility. D and F are outside the circle and therefore excluded.

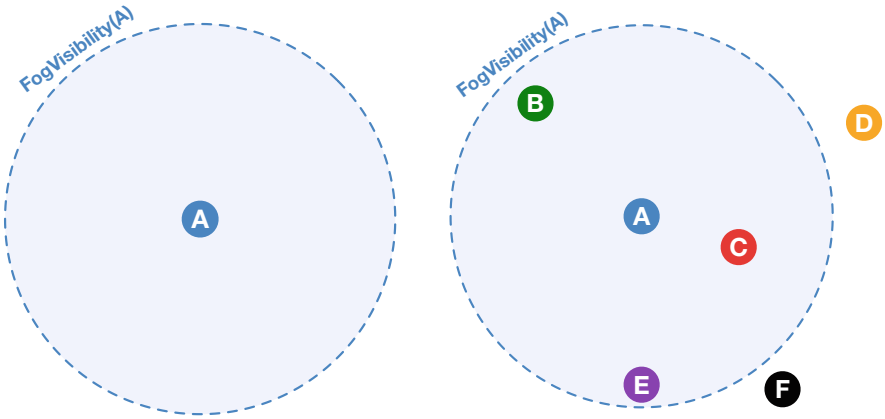

Fig. 2 Example graph of the Fog Visibility

2.3 *Fog Horizon*

The second concept is called Fog Horizon. Its name is on the one side inspired by the visual line separating the earth from the sky and on the other side by the event horizon of a black hole from which nothing can escape its gravitational field. Transferring those ideas to the area of Fog Components, it describes devices that closely interact with each other, have a certain locality attached, and share a common medium, in our case a communication channel.

This aspect can be described using the introduced Fog Visibility. As shown in Definition 2, the Fog Horizon is the symmetrical closure of the Fog Visibility and contains all Fog Components that can send as well as receive messages from each other, establishing a bidirectional communication. Based on this definition and the property of the Fog Visibility, the Fog Horizon is reflexive as well as symmetric. Accordingly, if one Fog Component A is in the Fog Horizon of Fog Component B, B is also in the Fog Horizon of A.

Definition 2: Fog Horizon

$$FogHorizon(x) := FogVisibility(x)^{\leftrightarrow} =$$
$$FogVisibility(x) \cap FogVisibility(x)^{-} =$$
$$\{y \mid y \in FogVisibility(x) \ \wedge \ x \in FogVisibility(y)\}$$

with:
$x, y \in FogComponentSet$

Figure 3 shows a Fog Horizon example: As the circle around A does not include any other Fog Components, its Fog Visibility and therefore Fog Horizon only contain the component itself. In the case of Fog Component B, these sets do not match up. While B can "see" D, this is not true the other way around, making the

Fog Visibility of B equal to B and D and the Fog Horizon just B. C and D on the other hand can send messages to each other, including them in each other's Fog Horizons.

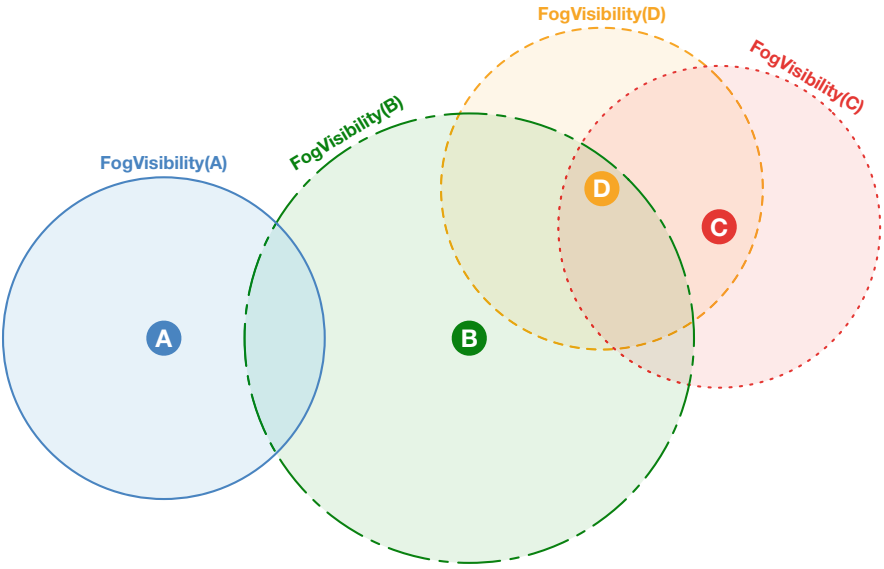

Fig. 3 Example graph for the Fog Horizon

2.4 *Fog Reachability*

The Fog Reachability describes the maximum outreach a Fog Component can have and therefore addresses the missing indirect communication from the Fog Visibility and Fog Horizon. While these concepts establish small isolated sets, the Fog Reachability also includes Fog Components that can be reached using other Fog Components as hops. Accordingly, as shown in Definition 3, the Fog Reachability is defined as the transitive closure of the Fog Visibility or every Fog Component that can receive direct or indirect messages.

Definition 3: Fog Reachability

$$FogReachability(x) := FogVisibility^{+}(x) =$$
$$\{y \mid y \text{ receives direct or indirect messages from } x\}$$

with:
$$x, y \in FogComponentSet$$

Figure 4 shows an example with four Fog Components A, B, C, and D. With A's Fog Visibility containing B and B's Fog Visibility containing C and D, A's Fog

Reachability contains all four Fog Components. B does not contain A, C does not contain any Fog Components and finally D contains B and C.

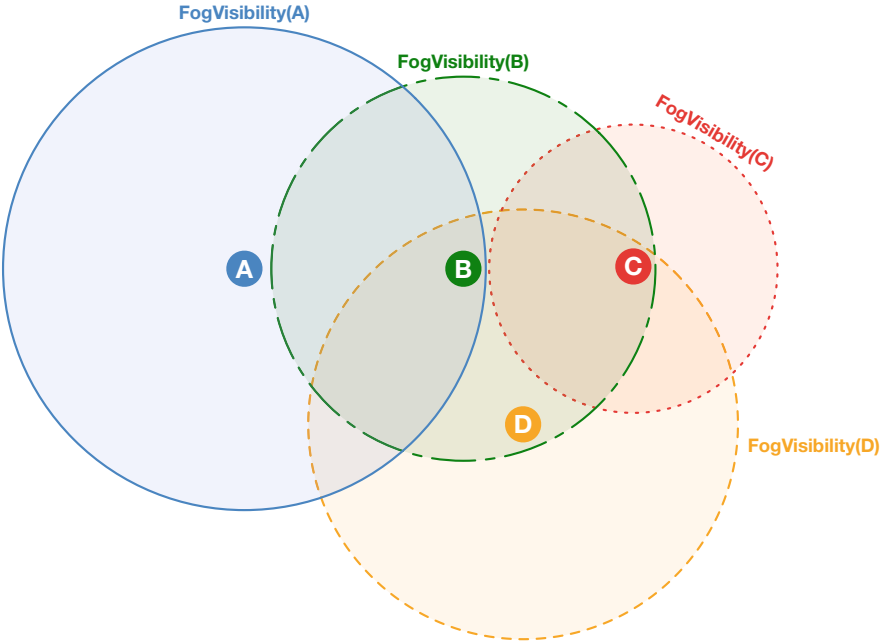

Fig. 4 Example graph for the Fog Reachability

One issue of the Fog Reachability is shown as soon as the Fog Components are not an isolated set, but open to the internet. According to the Fog Reachability's definition, just one open connection would result in a huge part of the internet being included in the Fog Reachability.

2.5 Fog Set

To address this issue, we have to further limit the Fog Components that are included in the set that describes a Fog Architecture, since we would not consider all these Fog Components to be part of a single architecture but rather a superset. Similar to the definition of the Fog Reachability, we use the transitive closure to create a set that is not limited to single hops, but this time for the Fog Horizon instead of the Fog Visibility as shown in Definition 4. This makes the Fog Set reflexive and symmetrical, but also transitive, and allows us to emphasize the close interaction between Fog Components that is described by the Fog Horizon.

Definition 4: Fog Set

$$FogSet(x) := FogHorizon^{+}(x) = \{y \mid y \in FogVisibility^{+}(x) \ \wedge \ x \in FogVisibility^{+}(y)\}$$

with:

$$x, y \in FogComponentSet$$

Figure 5 shows an example for such a Fog Set. Since A's Fog Horizon includes B and B's Fog Horizon includes C and D, the Fog Set of A includes all four Fog Components. In fact, this is also the case for the Fog Sets of the other Fog Components.

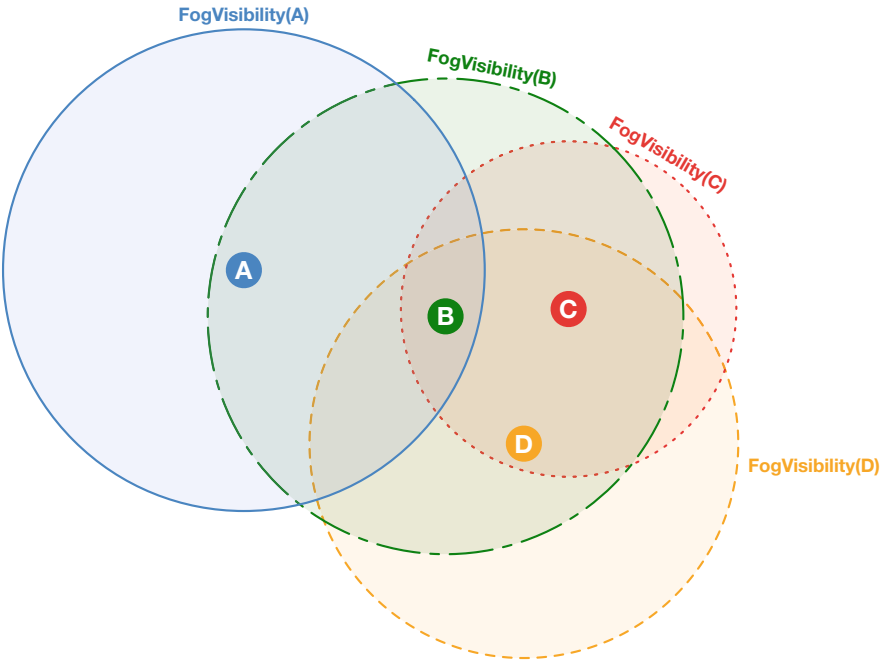

Fig. 5 Example graph for the Fog Set

Since the Fog Set is reflexive, symmetrical, and transitive, it is an equivalence relation and therefore the same and unique set for each involved Fog Component—which is what we would expect from Fog Components in the same Fog Architecture.

2.6 Service Constraints

In addition to the previously introduced concepts, most architectures are limited to specific services which are offered within them. These services are summarized in the *Service Set* as shown in Definition 5.

Definition 5: Service Set

$$ServiceSet := \{s \mid s \text{ is a Service } \}$$

To address this, the following three service sets—Provide, Consume, and Interest—create sets based on the Fog Components relation to the given service as defined by their names (Definition 6).

Definition 6: Service Sets

$$Provide(s) := \{x \mid x \textit{ offers and advertises } s \in ServiceSet\}$$
$$Consume(s) := \{x \mid x \textit{ requests } s \in ServiceSet\}$$
$$Interest(s) := Provide(s) \cup Consume(s)$$

with:
$x \in FogComponentSet$
$s \in ServiceSet$

Using these definitions, we define the following sets P, C, and I, which present selections on the given Fog Concept with respect to the provided service as shown in Definition 7.

Definition 7: Service Constraint

$$P_s(f(x)) := f(x) \cap Provide(s)$$
$$C_s(f(x)) := f(x) \cap Consume(s)$$
$$I_s(f(x)) := f(x) \cap Interest(s)$$

with:
$x \in FogComponentSet$
$f(x) \in \{ FogVisibility(x),\ FogHorizon(x),$
$\quad FogReachability(x),\ FogSet(x)\}$
$s \in ServiceSet$

These sets will be of particular interest in Sect. 3.1 as soon as we establish layers.

2.7 Communication Set

The second set we want to investigate is the *Communication Set*, and therefore the interactions/connectors between the different components of the Fog Set.

While most architectures use the same communication medium for the entire communication, Fog Architectures often involve different communication channels depending on the devices used. These channels can range from close proximity such as NFC or Bluetooth up to long distance communication such as 3G, 4G, or 5G. The following Fig. 6 shows an excerpt of different communication channels which can be found within Fog Architectures.

In the diagram, the channels are placed within a 2×2 grid in which the columns indicate whether the channels are wired or wireless and the rows indicate local and remote proximity. Therefore, the placement within the grid describes a double inheritance of the contained channel to the according superclasses. Additionally, all superclasses are `Communication Channels` themselves. This set, called *Communication Channel Set*, is defined in Definition 8. It contains all potential communication channels that can be used within a Fog Architecture.

Definition 8: Communication Channel Set

$CommunicationChannelSet = \{c \mid c \text{ is a communication channel.}\}$

While most communication channels that are used in the context of Fog Architectures are bidirectional, network limitations as well as sensors only providing data can include unidirectional communication channels. To include and address these channels, our definition of Fog Visibility is unidirectional in comparison to the Fog

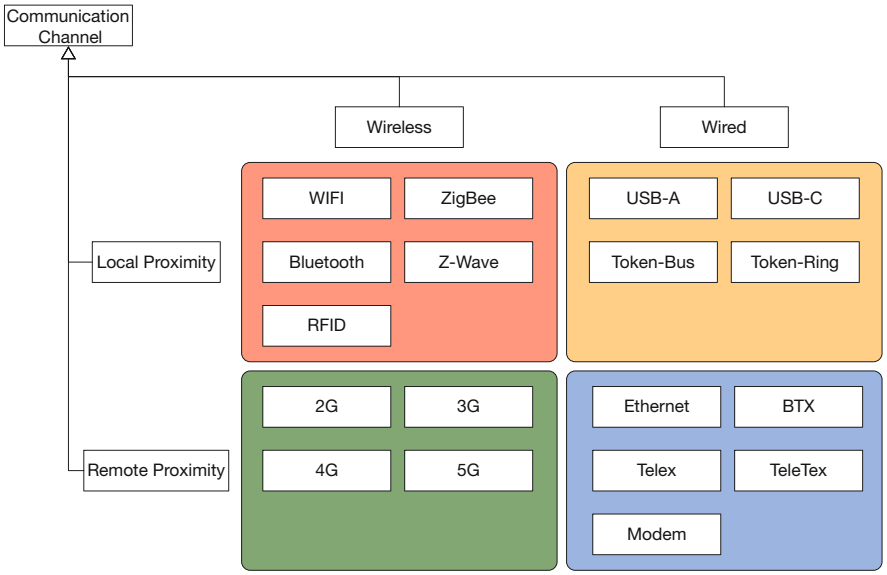

Fig. 6 Excerpt of possible communication channels in the context of Fog Architectures: The channels are divided based on the physical medium that they are using (wired versus wireless), but also the physical distance for which the communication channels can be used

Horizon, and accordingly, the Communication Set can also include unidirectional communication channels.

Bidirectional communication channels can be easily limited to unidirectional communication using software solutions. Channels such as Wi-Fi, Bluetooth, or USB rely on an open bidirectional communication approach. Therefore, encapsulating a certain amount of trust to other participants within the communication structure is essential for Fog Architectures; only trusted instances are described as part of a Fog Architecture.

In addition to the equivalent of a *Component* in the context of a Software Architecture which is the Fog Component and the related sets, we define a *Communication Component* as the equivalent to the *Connector*.

A *Communication Component* is a triple that consists of the source Fog Component, the used *Communication Channel*, and the destination Fog Component as shown in Definition 9. Specifying the source and destination allows to define unidirectional communications.

Definition 9: Communication Component

$$CommunicationComponent := \{SourceFogComponent, CommunicationChannel, DestinationFogComponent\}$$

All *Communication Components* of one Fog Architecture are grouped within the *Communication Set*. The *Communication Set* is defined as shown in Definition 10 and is the equivalent to the Fog Set. For a *Communication Component* to be considered part of the *Communication Set*, the *source* and *destination* Fog Component have to be part of the Fog Set, and the Communication *channel* has to be part of the *Communication Channel Set*.

Definition 10: Communication Set

$$CommunicationSet := \{c : CommunicationComponent = (source, channel, destination) \mid source \in FogSet \wedge destination \in FogSet \wedge channel \in CommunicationChannelSet\}$$

3 xFogPlus: Dynamic and Scalable Fog Architectures

xFogPlus addresses technical research goals 2 and 3 to achieve dynamic reconfigurability and scalability of Fog Architectures. This allows the addition of new components to existing Fog Architectures and the addition of layers.

3.1 Dynamic Reconfigurability

Being able to add new components to a Fog Architecture is simplified based on the overarching mathematical definitions introduced in Sect. 2. Based on those definitions, each component that should be considered part of the Fog Architecture needs to fulfill two requirements. First, the component needs to be part of the Fog Component Set, and second, it needs to be part of the Fog Set: As all concepts introduced in Sect. 2 are based on the Fog Component Set, it is a mathematical necessity for all of them, including the Fog Set, but the Fog Set is only sufficient for the Fog Component Set.

The Fog Component Set itself already poses a problem. Based on the definition shown in Sect. 2.1, the Fog Component Set consists of all potential Fog Components. While Fog Components can be any IoT device on MOF level M0 and M1, the Fog Component itself is an instance of the Fog Type on M3, which in turn has the subclasses Edge Device, Fog Node, and Cloud Device. Although this definition is helpful if different Fog Architectures are available and they should be differentiated between each other, it is the wrong way around if new components should be added: In order to be considered part of the Fog Component Set, the component needs to be a Fog Component, and therefore already part of a Fog Architecture which is not the case for new components.

Therefore, to be able to add components to the Fog Component Set, an alternative definition for a Fog Component is required which solely focuses on properties of the component itself. The first hard requirement is that every Fog Component is necessarily an IoT device. This means that a component needs to have the capability:

1. To be interconnected
2. To have the intention to share information across platforms
3. To be uniquely addressable
4. To have computational capabilities

Soft requirements are that the components:

1. Preferably use wireless communication
2. Have an interest in locality
3. Have general-purpose computational power
4. Which they offer as services to other components

This definition allows us to extend the Fog Component Set by new components which are not involved in any Fog Architecture, yet.

The second requirement is that the component satisfies the definition of a Fog Set and thus can be included in a Fog Architecture. Based on Definition 4, for a *Fog Component* x to be included in a *Fog Set* , the *Fog Component* needs to be in the transitive closure of the *Fog Horizon* of a *Fog Component* within the Fog Architecture that the *Fog Component* should be added to. Accordingly, it is mathematically sufficient for the *Fog Component* to be able to send and receive direct messages to and from any single *Fog Component* in the Fog Architecture.

After adding new components to the Fog Component Set and Fog Set, we have to address the issue on which layer the component is added. In Fog Computing, components can be assigned to three layers: the Edge Layer, the Fog Layer, or the Cloud Layer. While the Edge Layer and Cloud Layer consist of single layers, the Fog Layer can consist of several individual layers.

To indicate that the Fog Layer can consist of several layers, we rename the Fog Layer to *Fog Layer Set* in compliance with the introduced concept of xFogCore. The Fog Layer Set can consist of several Fog Layers itself.

Definition 11: Fog Layer Set

$FogLayerSet := \{l \mid l \text{ is a Fog Layer}\}$

We present definitions for the different types of layers. Definition 12 shows the properties of a *Fog Component* to be considered part of the *Edge Layer*. Each *Fog Component* x needs to be part of the *Fog Set* , thus, part of the *Fog Architecture* , and does not provide any services to other *Fog Components* , which is represented by not having any service s which makes *Fog Component* x part of its provide set.

Definition 12: Edge Layer

$$EdgeLayer := \{x \mid x \in FogSet \ \wedge \ \nexists s \in ServiceSet : x \in Provide(s) \}$$

The *Fog Layer* is defined as every *Fog Component* x that is, equal to the *Edge Layer*, part of the *Fog Set* and for which at least two services s_1 and s_2 exist, so that *Fog Component* x consumes one of the services and offers the other one, as shown in Definition 13. This describes the idea that *Fog Components* in the *Fog Layer* bring services of higher layers, for example, the *Cloud Layer*, closer to the *Edge Layer* but also do their own calculations.

Definition 13: Fog Layer

$$FogLayer := \{x \mid x \in FogSet \ \wedge \ \exists s_1, \ s_2 \in ServiceSet : \\ x \in Consume(s_1) \ \wedge \ x \in Provide(s_2) \}$$

The *Cloud Layer*, as shown in Definition 14, includes every *Fog Component* that does not consume any service itself.

Definition 14: Cloud Layer

$$CloudLayer := \{x \mid x \in FogSet \ \wedge \ \nexists s \in ServiceSet : x \in Consume(s) \}$$

Each *Fog Layer* in the Fog Layer Set is defined by a service pair s_i, s_j that is on the one side consumed by the layer and on the other side provided by the layer. If different layers include the same Fog Components although being defined by

different service pairs, those layers are fused to one layer consuming or providing several services.

Using the provided definitions for the different layers, the issue of adding Fog Components to specific layers can be reduced to these Fog Components providing or consuming the services that uniquely identify each layer.

Figure 7 shows an example of different Fog Components distributed on three layers: `Edge Layer`, `Fog Layer Set`, and `Cloud Layer`. The `Fog Layer Set` consists of three layers: `Fog Layer 1`, `Fog Layer 2`, and `Fog Layer 3`.

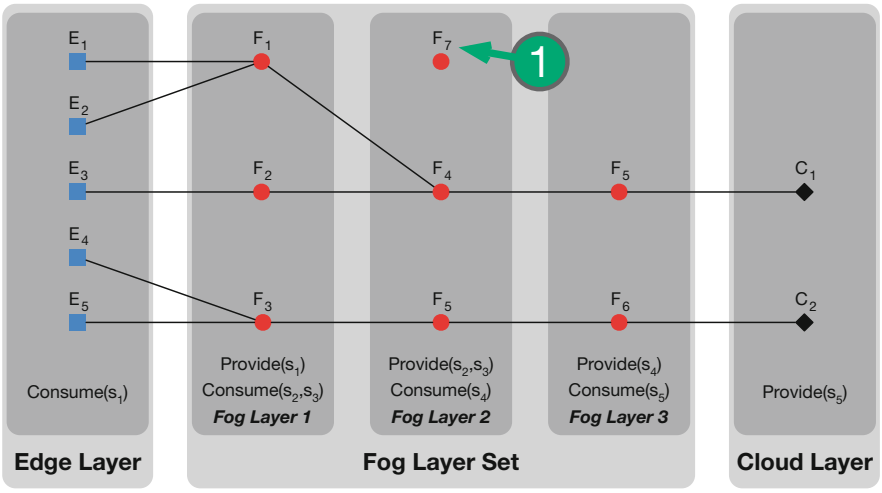

Fig. 7 The graph shows an example for a Fog Architecture which is distributed over the three layers *Edge Layer*, *Fog Layer Set*, and *Cloud Layer*

Although, in this example, all Fog Components in the `Edge Layer` consume the same service s_1, they could also consume other services as long as they do not offer any services themselves. The first two `Fog Layers` are examples for multiple consumed or provided services (s_2, s_3). These two layers present an instance of fused `Fog Layers` as the service pairs s_1, s_2 as well as s_1, s_3 result in the same set of Fog Components and therefore are on the same layer. The third `Fog Layer` provides service s_4 and consumes s_5 which is provided by the `Cloud Layer`.

Additionally, the example shows the case that `Fog Component` F_7 is added to the Fog Architecture. Based on the provided and consumed services s_2, s_3, and s_4, the new Fog Component can be added to the second `Fog Layer`; although no connections are established, yet.

3.2 Scalability

Adding new layers to existing architectures is one key aspect for making them dynamic and scalable. As most applications nowadays use client-server architectures, it is also essential for the transition from a centralized approach to a decentralized approach using Fog Computing.

With the introduced layer definitions and their service-based nature, the process of adding new layers to a Fog Architecture is simplified to the addition of a new Fog Component that offers and consumes a new pair of services that is used by existing Fog Components of the architecture.

Based on the position where the new layer should be added, we have to differentiate three cases. First, the addition of a layer between the layers of a client server architecture to create a Fog Architecture, second the addition of new Fog Layers in existing Fog Architectures, and third the creation of new Edge or Cloud Layers. Figure 8 shows three examples, one for each of these cases, with the previous architecture on the left and the resulting Fog Architecture on the right. Each example shows the resulting service configuration.

3.3 Handling Complexity

In order to handle the introduced complexity in Fog Architectures by new Fog Components and layers, and to make Fog Architectures more accessible for different stakeholders, we introduce a Fog Architecture view concept. A **View** of a Fog Architecture is a part of a Fog Architecture that consists of a specified amount of layers, which are of current interest for a stakeholder. The definition is based on the view concept introduced in SysML which provides a perspective that spans different abstractions, in our case layers [16]. Our *View* is defined based on a tuple that contains natural numbers which refer to the selected layers of interest.

Definition 15: Fog Architecture: View, Viewpoint, and Abstraction Level

$View(s) := \bigcup_{i=0}^{|s|-1} Layer(s_i)$

with:

$s :=$ Tuple containing the numbers referring to the selected layers
$|s| =$ Amount of layers within the View $\leq |FogArchitecture|$
$View \subseteq FogArchitecture$

Figure 9 shows a view example for a Fog Architecture describing a smart city with six layers. The view highlights the Edge Layer and the lowest two Fog Layers.

Looking at the View, one can describe the view as a Fog Architecture itself using the previous introduced layer definitions. Accordingly, Fog Components can take different rolls depending on the current View. For example, Fog Components on the "Street" layer which are on Fog Layer 2 of the entire Fog Architecture act as the Cloud for the highlighted View.

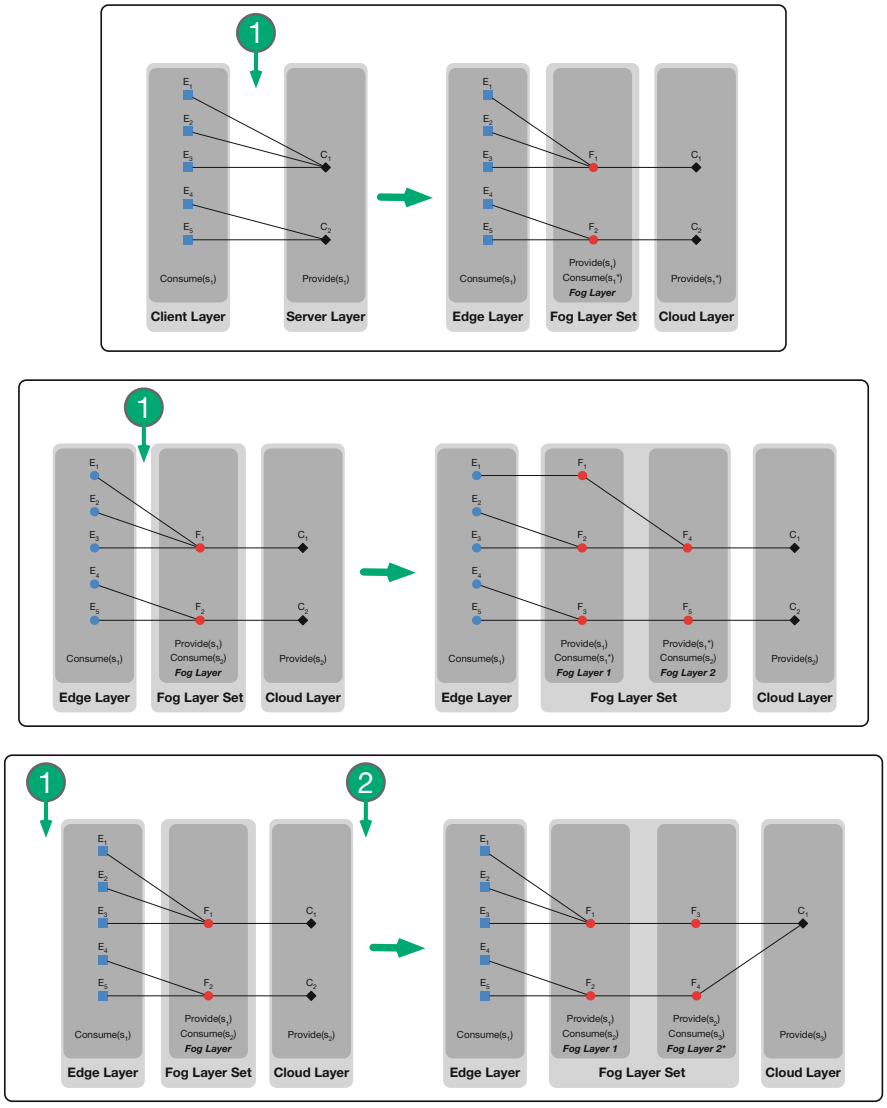

Fig. 8 Three examples of layer additions: the transition to a Fog Architecture, the addition of a new Fog Layer, and the addition of a new Edge/Cloud Layer

4 xFogStar: A Workflow for Service Provider Selection

Based on xFog, and accordingly xFogCore and xFogPlus, many new concepts can be established in dynamically, scalable Fog Architecture. We introduce xFogStar, one such concept, that focuses on the relation between service consumers and service providers, the players suggested by Bonomi et al. [8], to show an application

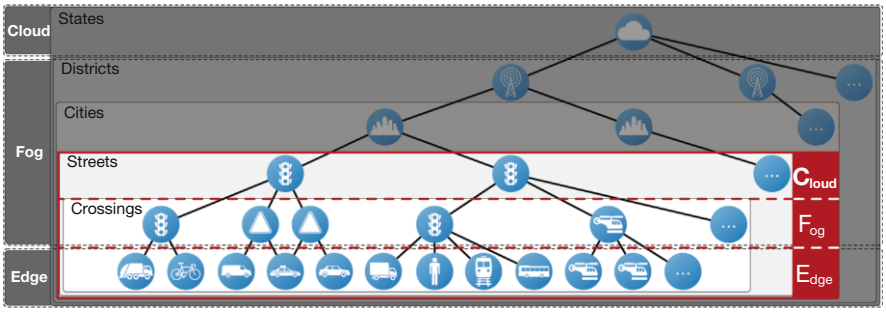

Fig. 9 View example of a Fog Architecture describing a smart city with six layers

of the introduced framework with a Fog Architecture containing two layer views. xFogStar supports the service provider selection in case multiple providers are within the Fog Horizon of a service consumer requesting one service, addressing technical research goal 4.

To describe properties of the service provider and its service, but also the requirements of the service consumers, we use QoS parameters and an according QoS vector. We investigated an extensive list of QoS parameters that are of major importance in Fog Architectures. Figure 10 provides an overview of all QoS parameters and groups them according to their dependencies.

Depending on the application domain, a selection of these parameters can be used to match the requirements of the service consumer with the service providers. This process to match the service consumer with the best fitting service provider for a specified service is depicted in Fig. 11. It shows a seven-step workflow which translates the requirements using availability strategies, limits, comparability strategies, ordering strategies, and weightings into a transparent, ordered list of service providers for the service consumer.

5 Validation

Due to the extent of the empirical validation of the xFog framework and xFogStar addressing *Knowledge Goal 2* and *Knowledge Goal 3*, this section only provides an overview of the conducted validation. Detailed descriptions on each individual case study, including each case study's design, its results, and the discussion, can be found in the dissertation "Dynamically Scalable Fog Architectures" [10].

The validation tries to justify if stakeholder goals would be met if the treatment is implemented in the problem domain's context. It investigates if the requirements for the treatment are addressed within a model of the problem domain. As the validation is part of the design cycle, and thus conducted in a laboratory setting, the implementation of the treatment in the problem domain is not of interest, yet. This results in the validation being independent of the stakeholders, which is the

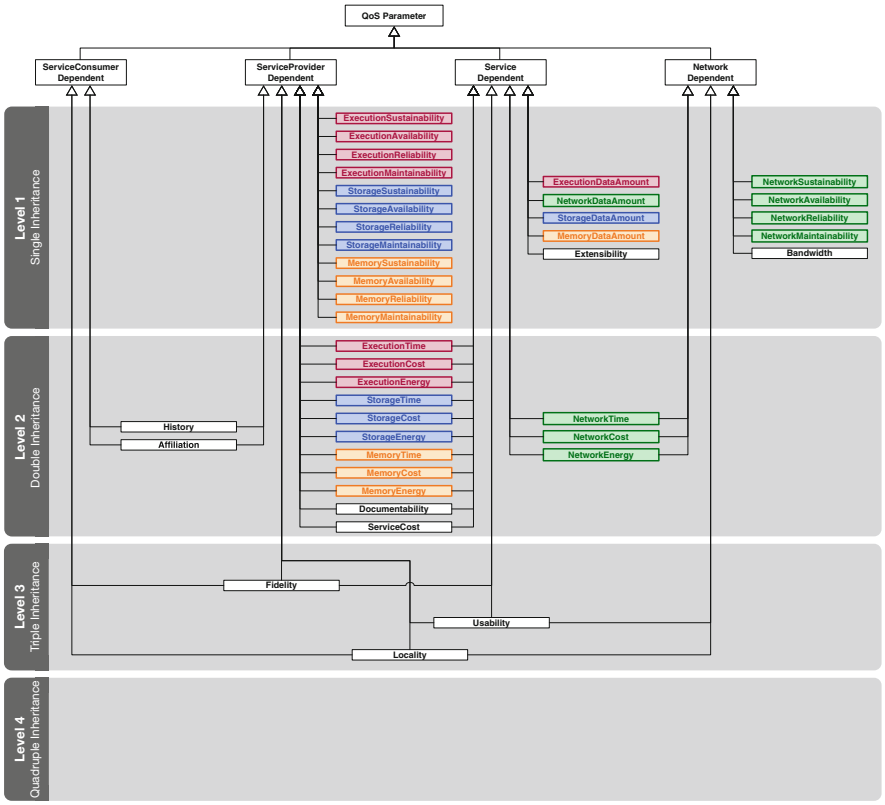

Fig. 10 Overview on QoS parameters for Fog Architectures and their dependencies

main difference between validation and evaluation. Therefore, different research approaches are used such as modeling, simulations, and testing [11].

The validation of xFog was separated into the three aspects of the xFog framework provided by xFogCore, xFogPlus, and xFogStar. Each validation relied on the validation approaches modeling and simulation. First, we introduced the design of the different case studies. We presented the problem domain of the case study, the requirements, and which concepts of xFog or xFogStar were addressed. We selected cases in different domains to support domain-independent conclusions. In total, we addressed six different domains: *Smart Environments*, *Smart Cities*, *Health*, *Continuous Integration*, *Metrology*, and *Gaming*. These domains were used in eight case studies mapped to three validations.

Second, we reported on the results of the validation for the three core concepts: Dynamic Fog Components, Scalable Fog Architecture, and Service Provider Selection. While the formalization of Fog Computing (xFogCore) is used throughout all three concepts, each of the concepts can be assigned to an addition to xFog as shown in Fig. 12. *Dynamic Fog Components* and *Scalable Fog Architectures* are covered

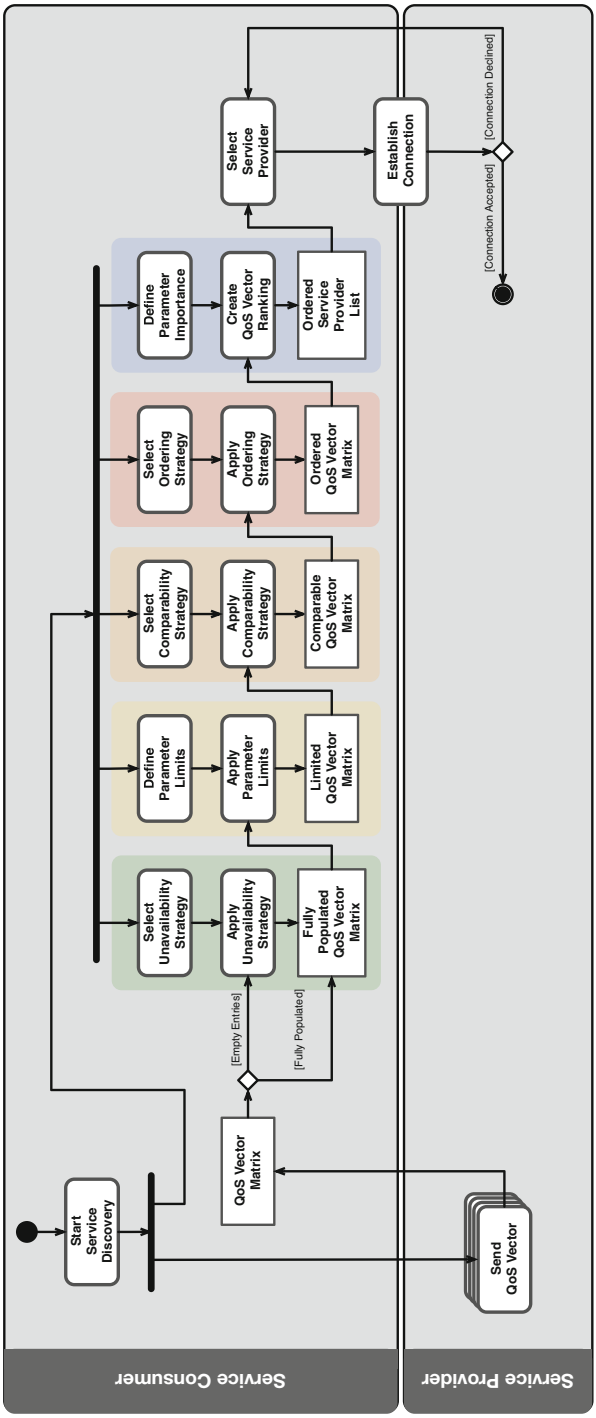

Fig. 11 The workflow for service provider selection (UML Activity Diagram)

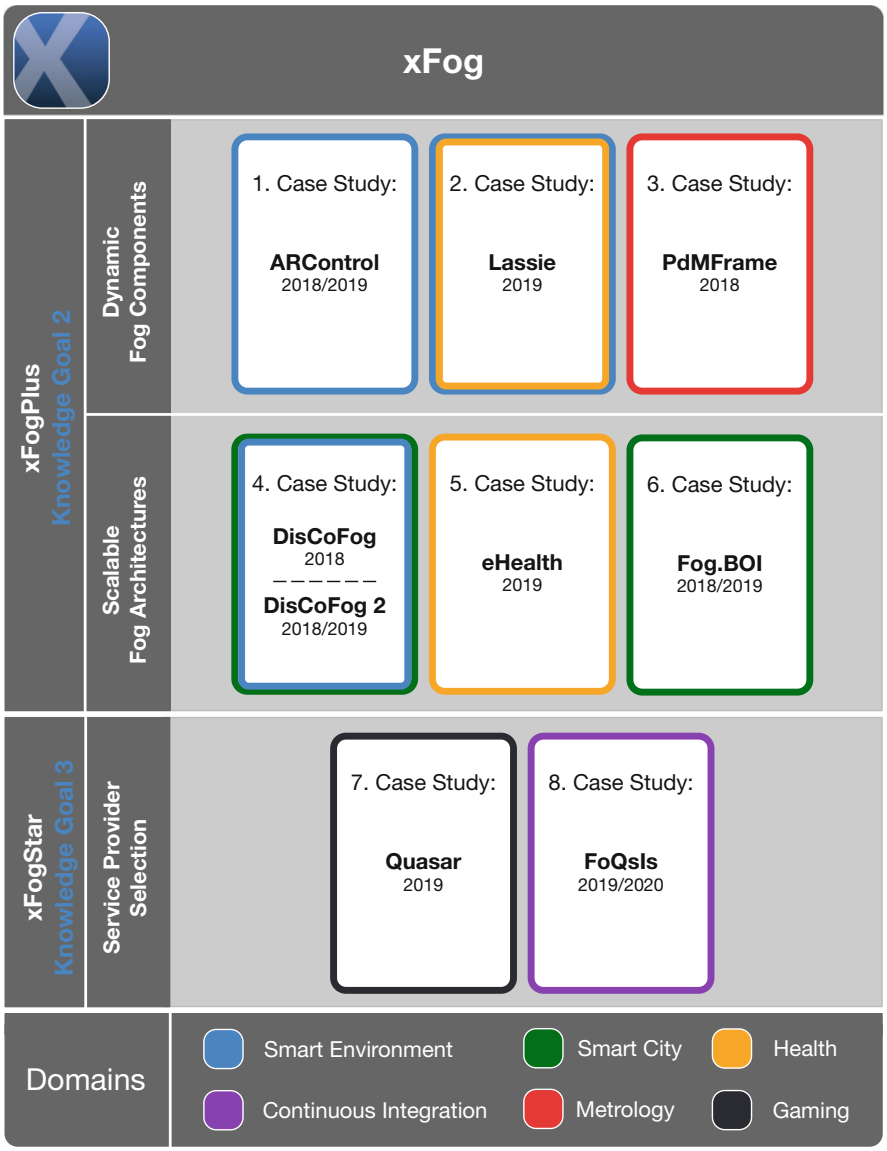

Fig. 12 An overview of the validation design for the xFog framework. Each of the three rows represent a validation with Dynamic Fog Components and Scalable Fog Architectures belonging to concepts related to xFogPlus and the Service Provider Selection belonging to the xFogStar workflow. The coloring of the case studies represents the domains they belong to

in xFogPlus as shown in the first two rows of the diagram. The *Service Provider Selection* uses xFogCore and the workflow introduced by xFogStar.

Third, we discussed the impact of the results for the xFog framework, interpreted the results, and addressed threads to the validity of the validation approach.

6 Conclusion

The main goal of the dissertation "Dynamically Scalable Fog Architectures" [10] was to create a framework that establishes a formalization for Fog Computing and integrates support for mobile applications and dynamic reconfigurability of Fog Architectures. We implemented the formative research approach design science using treatment designs and treatment validations.

The results can be separated into three parts: The problem investigation and description, the creation of the xFog framework and its validation. The problem investigation resulted in the goals shown in Fig. 1, which we translated into use cases and functional and nonfunctional requirements. These were addressed by the xFog Framework which can be divided in xFogCore, xFogPlus, and xFogStar.

xFogCore defined the *Component Set* represented by the Fog Set and the *Communication Set* which relate to the components and connectors of a software architecture. To define the Fog Set, xFogCore introduced the *Fog Component Set*, *Fog Visibility* , *Fog Horizon* , and *Fog Reachability* . These concepts describe the components of a Fog Architecture based on mathematical definitions. We showed how the component sets can be constrained to specific services that are offered or consumed, or that are of interest for a Fog Component, which allows the identification of layers within Fog Architectures. We defined the *Communication Set* as a set of *Communication Components* which are defined by the involved Fog Components and the used communication channel. The sets were put into context by a meta model on MOF level M2 including the basic building blocks of software architectures which allowed the interpretation of the sets as graphs.

xFogPlus introduced support for the dynamic addition of Fog Components to the Fog Architecture at runtime by redefining the idea of Fog Components and by providing definitions for the three layers: *Edge Layer*, *Fog Layer*, and *Cloud Layer*. Second, new layers can be described and added to the Fog Architecture enabling scalability. As the scalability increased the complexity of the Fog Architecture, we established the concept of different *Views* on the Fog Architecture to set a focus on different layers depending on the stakeholder's current interest.

xFogStar defined a workflow for the service provider selection in dynamically scalable Fog Architectures which are described by the concepts of xFogCore and xFogPlus. The workflow is used to select the best fitting service provider for the service consumer's needs. These needs are represented as a vector of QoS parameters which we defined and categorized according to their dependencies. We investigated the different steps of the xFogStar workflow to outline arising problems.

We validated three aspects that use the foundations and formalization of xFog to investigate *Knowledge Goal 2* and *Knowledge Goal 3*: *Dynamic Fog Components* , *Scalable Fog Architectures* , and the *Service Provider Selection*. For each aspect, we used a multiple case study to gather quantitative data on the feasibility of xFog, and thus xFogCore, xFogPlus, and xFogStar. *Dynamic Fog Components* and *Scalable Fog Architectures* related to xFogCore and xFogPlus, while the *Service Provider Selection* addressed the xFogStar workflow.

Each validation compared the expected results with the results provided by xFog. The first validation investigated three cases from different domains to support generalizable conclusions. It demonstrated the feasibility of xFog and in particular xFogPlus by examining the addition of components at runtime. The second validation included three cases to show the feasibility of the scalable concepts of xFogPlus by adding new layers to existing Fog Architectures. The resulting Fog Architectures were used to highlight the applicability of the *View* concept which addresses complexity depending on the stakeholder's current point of interest. The last validation for the *Service Provider Selection* demonstrated the feasibility of the xFogStar workflow in two cases.

Acknowledgments I would like to thank my first and second advisors Prof. Dr. Bruegge and Prof. Dr. Lichter for their valuable insights and feedback. Additionally, I would like to thank all members of the chair for applied software engineering from the Technical University of Munich for the discussions and the students who took part in my case studies.

References

1. Tanenbaum, A., Van Steen, M.: Distributed Systems: Principles and Paradigms. Prentice-Hall, Englewood Cliffs (2013)
2. Ferrer, A.J., Marquès, J.M., Jorba, J.: Towards the decentralised cloud: survey on approaches and challenges for mobile, ad hoc, and edge computing. ACM Comput. Surv. **51**, 111 (2019)
3. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**, 14–23 (2009)
4. Dinh, H., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mobile Comput. **13**, 1587–1611 (2013)
5. Qi, H., Gani, A.: Research on mobile cloud computing: review, trend and perspectives. In: 2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), pp. 1–6 (2012)
6. Satyanarayanan, M.: The emergence of edge computing. IEEE Comput. **50** 30–39 (2017)
7. Dolui, K., Datta, S. K.: Comparison of edge computing implementations: fog computing, cloudlet and mobile edge computing. In: Global Internet of Things Summit (GIoTS), pp. 1–6 (2017)
8. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog Computing and its role in the Internet of Things. In: MCC Workshop on Mobile Cloud Computing, vol. 1, pp. 13–16 (2012)
9. Marín-Tordera, E., Masip-Bruin, X., García-Almiñana, J., Jukan, A., Ren, G., Zhu, J.: Do we all really know what a fog node is? Current trends towards an open definition. Comput. Commun. **109**, 117–130 (2017)
10. Henze, D.: Dynamically Scalable Fog Architectures. Technische Universität München, München (2020)

11. Wieringa, R.: Design Science Methodology for Information Systems and Software Engineering. Springer, Berlin (2014)
12. Hevner, A., March, S., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
13. Shaw, M., Garlan, D.: Software Architecture. Prentice-Hall, Englewood Cliffs (1996)
14. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley, Reading (2003)
15. Kruchten, P.: The 4+ 1 view model of architecture. IEEE Softw. **12** 42–50 (1995)
16. ISO: IEC/IEEE systems and software engineering: architecture description. In: ISO/IEC/IEEE 42010: 2011 (E)(Revision of ISO/IEC 42010: 2007 and IEEE Std 1471-2000). IEEE, New York (2011)
17. Henze, D., Schmiedmayer, P., Bruegge, B.: Fog horizons—a theoretical concept to enable dynamic fog architectures. In: IEEE/ACM International Conference on Utility and Cloud Computing, vol. 12, pp. 41–50 (2019)